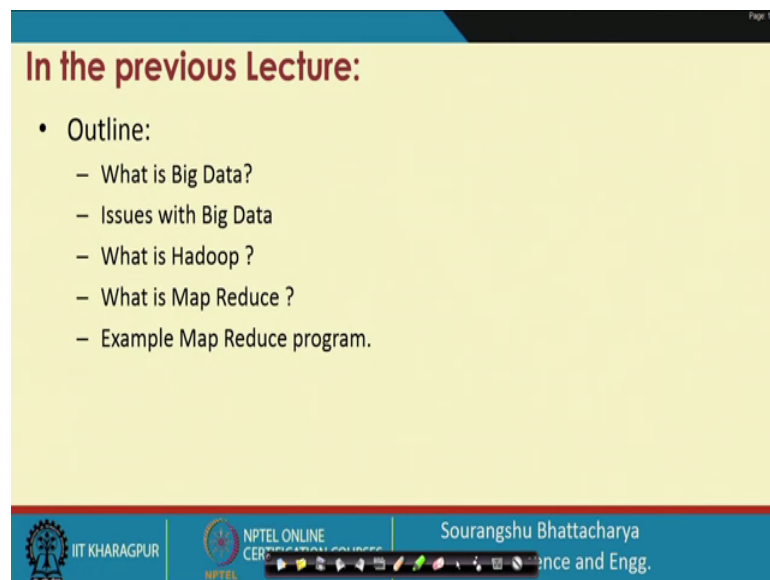


Scalable Data Science
Prof. Sourangshu Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 18a
Hadoop System

Hello everyone. Welcome to the 18th lecture of the Scalable Data Science course on NPTEL. I am Professor Sourangshu Bhattacharya from Computer Science and Engineering department, IIT Kharagpur. Today's lecture is going to be on Hadoop system. So, it will be the first lecture in a three part lecture on the Hadoop system.

(Refer Slide Time: 00:43)



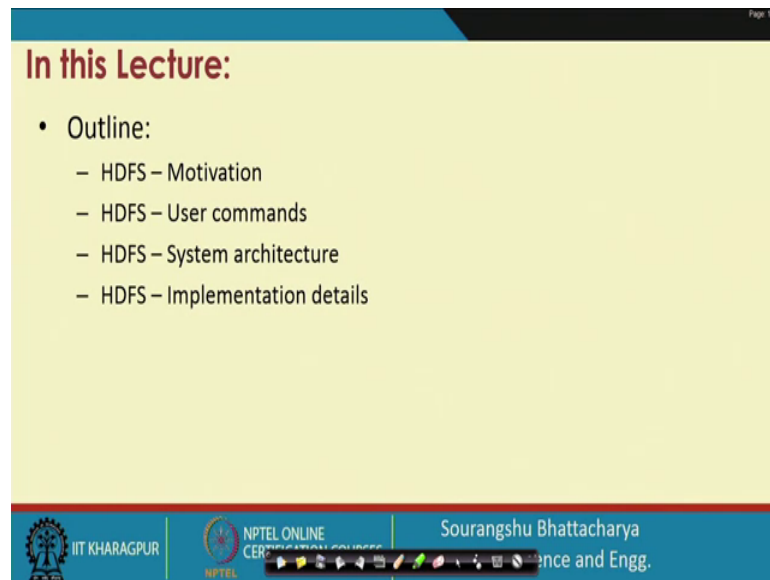
In the previous Lecture:

- Outline:
 - What is Big Data?
 - Issues with Big Data
 - What is Hadoop ?
 - What is Map Reduce ?
 - Example Map Reduce program.

The slide footer contains the IIT Kharagpur logo, NPTEL ONLINE logo, and the name Sourangshu Bhattacharya, Department of Computer Science and Engineering.

So, just to recall, in the previous lecture, we have seen what is big data and what is the motivation for big data processing. We have seen the issues with big data. What are the issues while you are trying to process big data? Then we have seen this system Hadoop and we have seen the programming paradigm which is called the map reduce paradigm. And we have seen some a few examples of map reduce programs and we have seen how to express a particular task in the basic map reduce framework.

(Refer Slide Time: 01:29)



Page 1/1

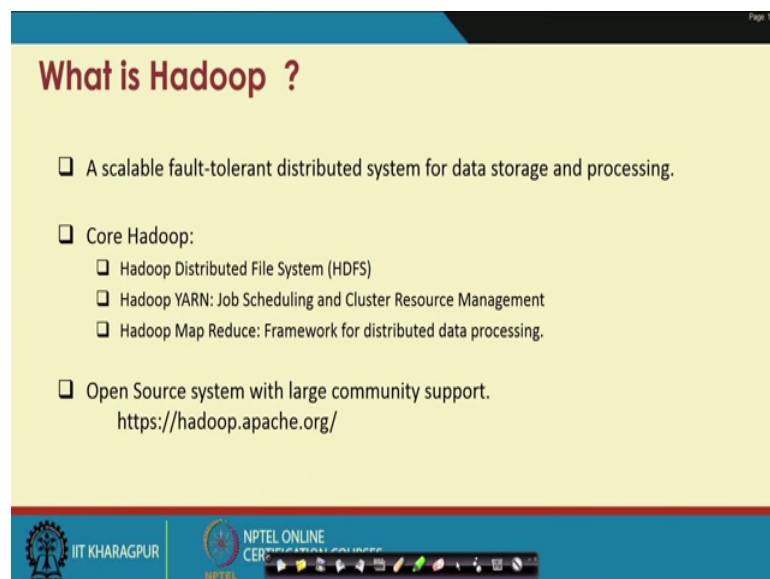
In this Lecture:

- Outline:
 - HDFS – Motivation
 - HDFS – User commands
 - HDFS – System architecture
 - HDFS – Implementation details

IIT KHARAGPUR | NPTEL ONLINE COURSE | Sourangshu Bhattacharya
Computer Science and Engg.

So, in this lecture, we will look at one of the components of Hadoop system, which is the Hadoop distributed file system ok. So, first we will discuss what are the motivations behind motivations and characteristics of this Hadoop distributed file system. Then we will see some of the commands that you can use and basically get a feel of how to interact with the Hadoop distributed file system. Then we will get into the system architecture of HDFS and also we will see a few implementation issues and implementation details of the HDFS ok.

(Refer Slide Time: 02:16)



Page 1/1

What is Hadoop ?

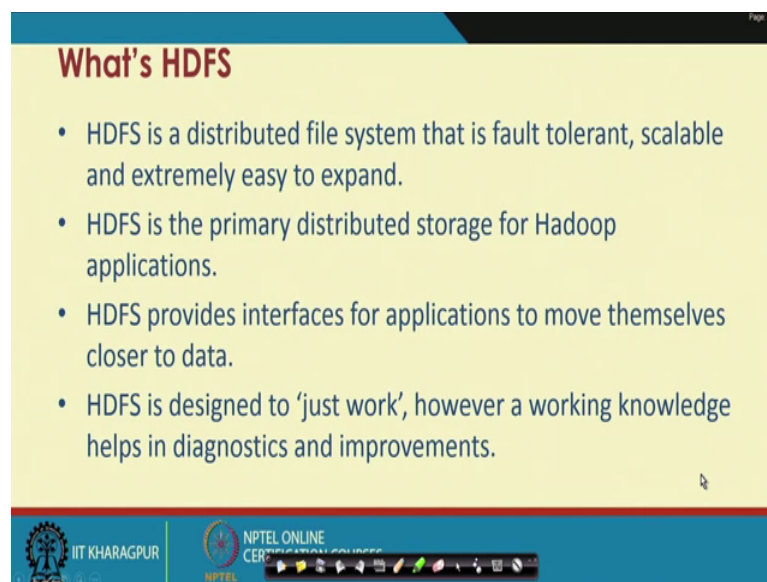
- A scalable fault-tolerant distributed system for data storage and processing.
- Core Hadoop:
 - Hadoop Distributed File System (HDFS)
 - Hadoop YARN: Job Scheduling and Cluster Resource Management
 - Hadoop Map Reduce: Framework for distributed data processing.
- Open Source system with large community support.
<https://hadoop.apache.org/>

IIT KHARAGPUR | NPTEL ONLINE COURSE | Sourangshu Bhattacharya
Computer Science and Engg.

So, just to recall, what is Hadoop, so Hadoop is a scalable fault-tolerant and distributed system for data storage and processing. So, as discussed earlier, the core Hadoop has three very important components. The first is the Hadoop distributed file system component, which is responsible for storing data in a distributed manner and also making this data available.

The second component is the Hadoop YARN component, which is responsible for scheduling jobs in the Hadoop cluster and also it is responsible for managing the resources of the cluster. And finally, there is the Hadoop map reduce framework, which we have distributed which we have discussed in the last class ok.

(Refer Slide Time: 03:12)



What's HDFS

- HDFS is a distributed file system that is fault tolerant, scalable and extremely easy to expand.
- HDFS is the primary distributed storage for Hadoop applications.
- HDFS provides interfaces for applications to move themselves closer to data.
- HDFS is designed to 'just work', however a working knowledge helps in diagnostics and improvements.

IIT KHARAGPUR | NPTEL ONLINE
CERTIFICATE COURSE

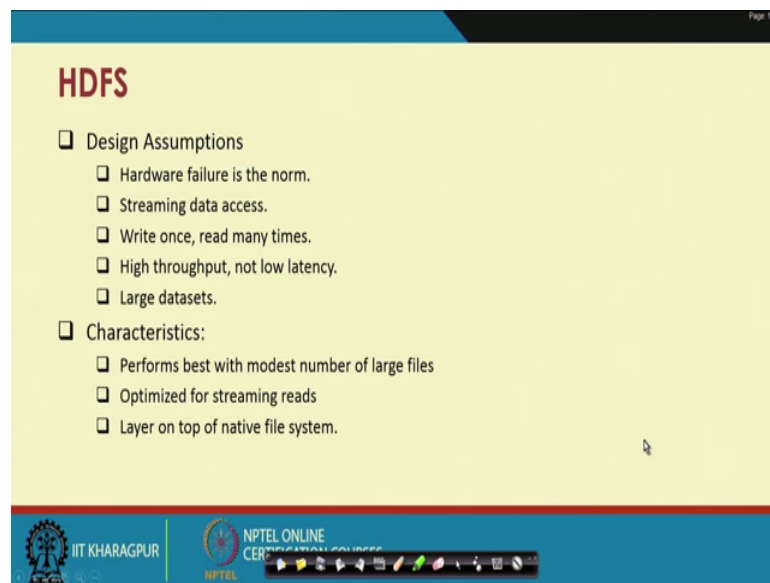
So, so what is HDFS or Hadoop distributed file system. So, HDFS is a distributed file system with this characteristics that it is fault tolerant, scalable and very easy to expand or extremely easy to expand ok. And HDFS is used as the primary distributed storage for any Hadoop applications ok. So, when you are writing an Hadoop application by default, you will be using HDFS as the main distributed storage for this applications.

Now, one of the main difference from HDFS to other distributed file systems is that it provides interfaces for applications to move themselves closer to data. As we had discussed earlier one of the main issues of big data processing is that we waste a lot of time moving data between the different nodes that we are processing it in. So, paradigm or idea is to instead of moving the data to where the computation is going on, you can

move the computation closer to data. And we will see how Hadoop distributed file system helps in that.

Then Hadoop distributed file system is designed to just work that is you need not know the internal functioning or how exactly it is designed. But, if you know the design and internal functioning, then you can better utilise the Hadoop distributed file system and also you can better debug the programs that use a Hadoop distributed file system.

(Refer Slide Time: 05:10)



So, what are the design assumptions behind the Hadoop distributed file system. So, the first design assumption is that hardware failure is a norm. So, basically all the data that you store in Hadoop distributed file system has to be tolerant to hardware failure that is at any point in time particular disc that you write the data in may fail and hence it should have a backup of the data. So, this is the first assumption.

The second design assumption is streaming data access. So, data will be access in a streaming manner one after the other one record after the other. So, this is true for also other standard file systems, but this is especially true for HDFS, since the speed of reading the data is of essential. So, this is one of the assumptions.

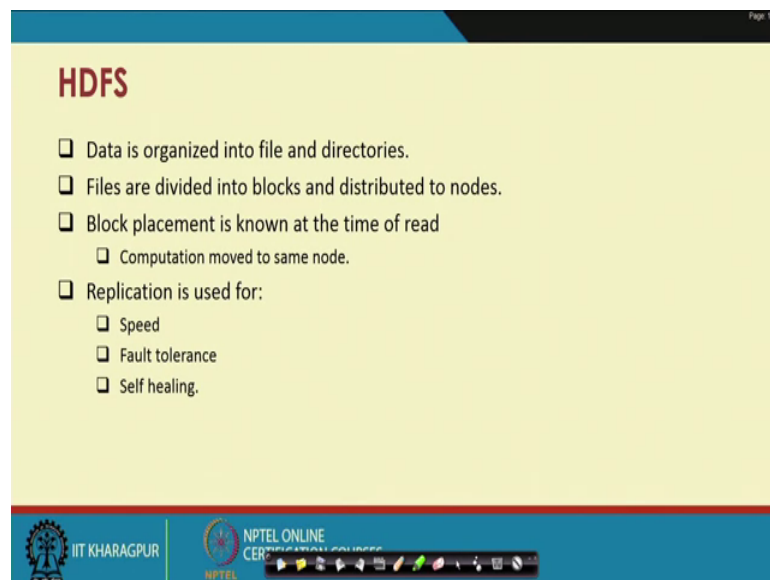
The other assumption is that is the right ones and read many times assumption. So, here the main idea is that since you are writing such big data files, it is expected that you will write this big data ones and then use it many times. Otherwise, you have not really you

know taken a lot of advantage of this big data, because you have spent so much effort gathering and then writing this big data. It is expected that once you write this, you will read it many types and use it many times.

The next assumption is that it is a high throughput, not low latency system. So, what this means is that if you if you try to access randomly data or if you try if your application is designed in such a manner that you are making many random access to different files or even different parts of the same file, then your application may run slower, as opposed to if you make a high throughput read of the whole data. And then it is designed for large data set, which is obviously the case.

So, keeping these design assumptions in mind. HDFS performs best with a modest number of large files. It is optimised for streaming reads as we discussed. And another important characteristic of HDFS is that it does not try to implement the low level file system, but it is a layer on top of the existing file system, which is provided by the operating system ok.

(Refer Slide Time: 08:23)



The slide is titled "HDFS" in a large, bold, red font. Below the title is a list of bullet points, each preceded by a square checkbox. The list includes: "Data is organized into file and directories.", "Files are divided into blocks and distributed to nodes.", "Block placement is known at the time of read", which has a sub-bullet "Computation moved to same node.", and "Replication is used for:", which has three sub-bullets: "Speed", "Fault tolerance", and "Self healing.". The slide has a yellow background and a blue header and footer. The footer contains the logos for IIT KHARAGPUR and NPTEL ONLINE, along with a navigation bar.

- Data is organized into file and directories.
- Files are divided into blocks and distributed to nodes.
- Block placement is known at the time of read
 - Computation moved to same node.
- Replication is used for:
 - Speed
 - Fault tolerance
 - Self healing.

So, what are the details? So, the first thing is that like any other file system even in Hadoop distributed file system, the data is organized into files and directories. And just as you as you work with any a unique distributed files, you can also work with files and directories in the Hadoop distributed file systems.

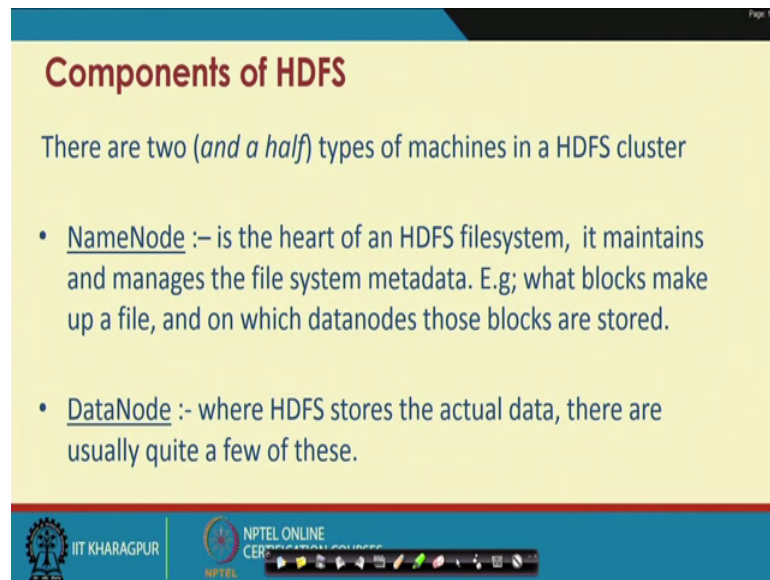
The now, also similar to standard file systems; files are divided into blocks, but now unlike standard file systems, these blocks are distributed to different nodes in a Hadoop cluster. So, these blocks are data that is sitting on the native file system of different nodes of the Hadoop cluster.

At the time of reading, the block placement is known, which means that you know the node on which a particular block is placed and hence you can so you can schedule your jobs or your map reduce tasks in such a manner that the computation goes to the data that is your application or the client that is reading the data is close to the node, where the data is being read.

Now, an important concept in Hadoop distributed file system is the replication concept ok. So, every or yeah every block is replicated a certain number of times. Now, this replication is useful for first of all speed of computation, because for example, if a particular block is placed is replicated across three different servers, we can schedule the computation close to one of those servers. And then the chance of finding a free let us say compute slot in one of the three servers is higher than if the data was not replicated, it was only on one server, then the chance of finding a free compute slot is lower.

The second important gain of replication is the fault tolerance that is if one of the server goes down, then the you can run the job on the other server without losing the whole job ok. And the third is it can be used for self healing that is once a particular node goes down ok, so then since the data is already replicated on other nodes, the those nodes can again heal that heal the file system. In the sense that they can replicate the data onto a fourth node. And hence, even though one node is down, the total number of replicated factors is still three.

(Refer Slide Time: 12:16)



The slide is titled "Components of HDFS" in a bold, dark red font. Below the title, it states "There are two (*and a half*) types of machines in a HDFS cluster". A bulleted list follows, describing the NameNode and DataNode. The slide footer includes the IIT Kharagpur logo, the NPTEL ONLINE logo, and a navigation bar with various icons.

Components of HDFS

There are two (*and a half*) types of machines in a HDFS cluster

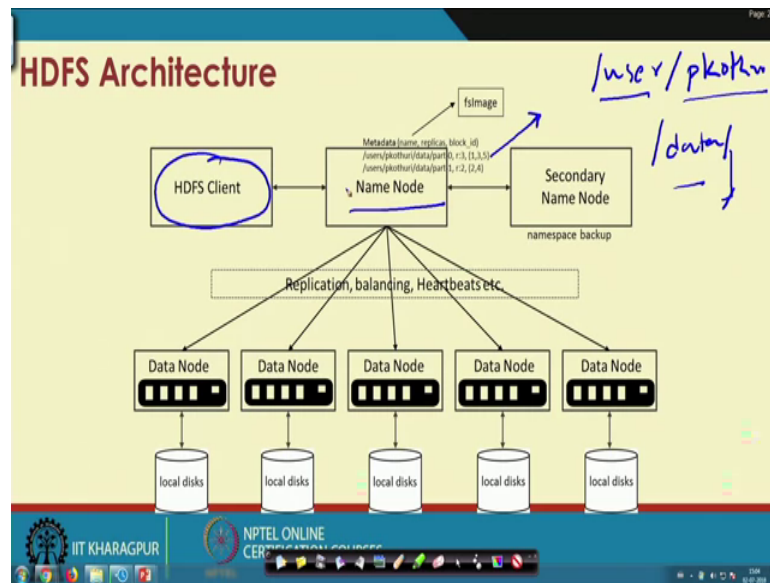
- NameNode :- is the heart of an HDFS filesystem, it maintains and manages the file system metadata. E.g; what blocks make up a file, and on which datanodes those blocks are stored.
- DataNode :- where HDFS stores the actual data, there are usually quite a few of these.

IIT KHARAGPUR | NPTEL ONLINE | CER

So, there are two main components of the HDFS file system ok. So, the first component and these are both these are software components, which are running on the servers. So, the first software component is the name node component. So, this is the heart of an HDFS file system, it maintains and manages all the file system metadata. So, we will look at what metadata it manages, but mainly for example, for a given file, it stores which blocks constitute these files and on which nodes these blocks are stored etcetera.

The second important component is the data node component ok. So, this is where HDFS stores the actual data and almost you can think that any Hadoop cluster almost all the nodes will work as a datanode, whereas there will be one or two name nodes. Typically, there will be only one primary namenode and then maybe a few secondary name nodes, which become useful, when the if the primary name node goes down.

(Refer Slide Time: 13:39)



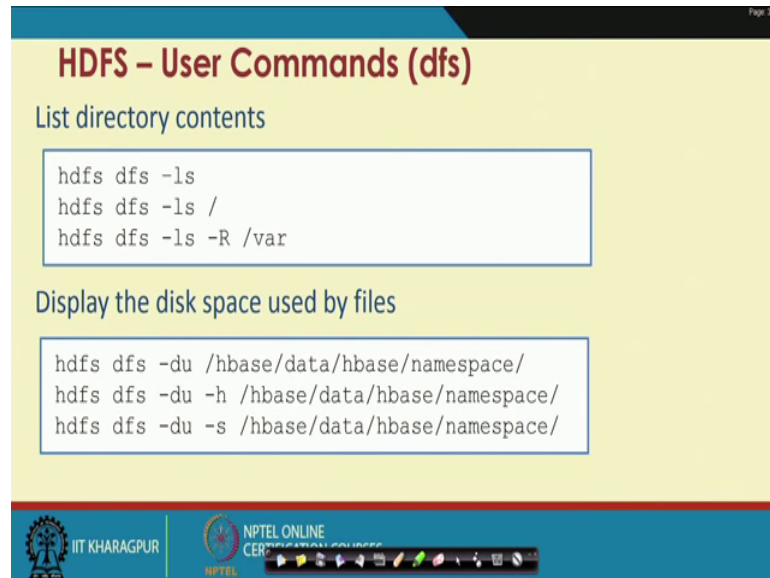
So, this is an example of a HDFS architecture you can see that. So, as you can see, so this is an example of HDFS URL, so slash user slash pkothuri slash data and then there will be part file. So, slash user slash pkothuri slash data is the location and then under this location there are this part files, which are stored. So, this is a very typical structure of a HDFS file system. So, there is the first three are the directories. So, there is the user directory, under which there is this directory and under which there is this directory and so on and so forth.

So, as you can see the components here are. So, there is the name node which is storing all this meta data, so which is storing the filename and it is storing the block id's etcetera. Now, this is the important is the most important thing, it is called the HDFS client. So, the HDFS client is the software that is programmed by you. So, for example, HDFS client could run inside a mapper job that the you as a programmer have defined or it could it could also one could also used inside a normal program, not a map reduce program per say but inside a normal program and HDFS client.

Now, the HDFS client should typically interact first with the name node to get the details of the data node, where the data is stored or where the blocks is stored. And then it would also interact with these data nodes to directly get the data. So and these data nodes are as you can see here running on almost all the machines in the Hadoop cluster, which are connected to their local disks and these are stores for the block. And the data nodes and

the name node, they interact with each other to provide various functionalities like replication, balancing, heartbeat etcetera. So, they basically perform all the function ok.

(Refer Slide Time: 16:45)

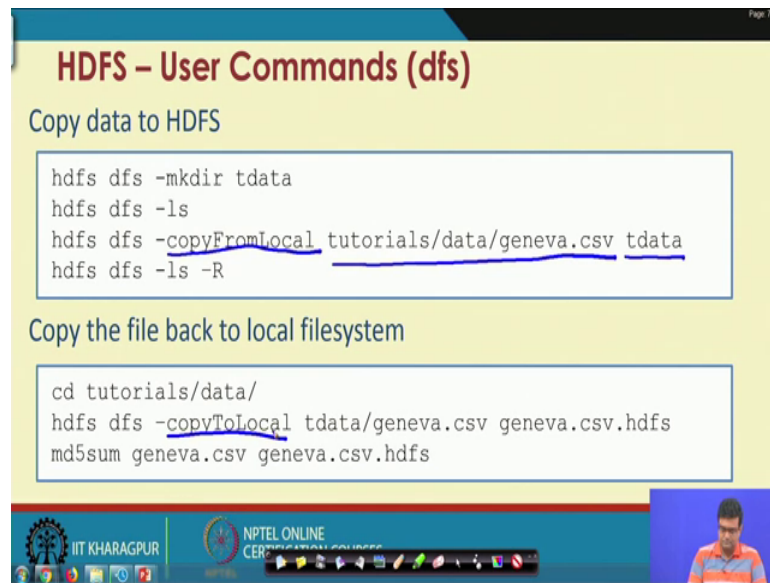


The slide is titled "HDFS - User Commands (dfs)" and is divided into two sections. The first section, "List directory contents", shows three commands: `hdfs dfs -ls`, `hdfs dfs -ls /`, and `hdfs dfs -ls -R /var`. The second section, "Display the disk space used by files", shows three commands: `hdfs dfs -du /hbase/data/hbase/namespace/`, `hdfs dfs -du -h /hbase/data/hbase/namespace/`, and `hdfs dfs -du -s /hbase/data/hbase/namespace/`. The slide footer includes the IIT KHARAGPUR logo, the NPTEL ONLINE CERTIFICATE COURSE logo, and a navigation bar with various icons.

So, this is just a view of few of the commands that you can use in HDFS. So, for example, you could use the list command just as in as a Unix command, so you can write `hdfs space dfs -ls` and this would list the contents of a current directory. So, whenever you are you are logged into a Hadoop cluster as a Hadoop user, you will have a current directory. So, Hadoop will maintain a current directory. And then you can you can use this command to view the contents. Similarly, you can use other commands to view for example, contents of root directory the second command or recursively view the contents of a particular directory.

Now, similarly, so just so you can use the `du` command to view the disk space occupied by various directories. So, for example, if you give a subdirectory, you could get that the you could it would list all the sub files in this directory and then it would list the space occupied by those files or directories in this directory, or it could give you just a summary, if you use the minus `s` flag or and it could give in a human readable format also with just the minus `h` flag. So, commands are pretty much similar to a UNIX command, so you can use it as if you are a UNIX file system.

(Refer Slide Time: 18:25)



The slide is titled "HDFS - User Commands (dfs)" and is divided into two sections. The first section, "Copy data to HDFS", contains a terminal window with the following commands: `hdfs dfs -mkdir tdata`, `hdfs dfs -ls`, `hdfs dfs -copyFromLocal tutorials/data/geneva.csv tdata`, and `hdfs dfs -ls -R`. The second section, "Copy the file back to local filesystem", contains a terminal window with the following commands: `cd tutorials/data/`, `hdfs dfs -copyToLocal tdata/geneva.csv geneva.csv.hdfs`, and `md5sum geneva.csv geneva.csv.hdfs`. The slide footer includes the IIT Kharagpur logo, NPTEL ONLINE CERTIFICATE PROGRAM logo, and a small video feed of a presenter.

Similarly, you can use for example, `mkdir` which is similar to the `make directory` or `mkdir` command in UNIX. And then, so one of the important tasks is to copy data from the local disk or So, you are logged in to a Hadoop server with the with an account to one of the machines ok and in the local file system of this machine, since you have an account, you may have some data.

So, let us say this data is called this tutorials data geneva dot csv, then you can use the `copy from local` command ok and you can give the name of the directory in which you want to copy. So, this directory should exist. So, you have already created this directory and now you can copy the data there. And similarly, once you are done with all the processing, you may want to copy the data from the distributed file system back to the local file system. And for this, you can use this `copy to local` command.

(Refer Slide Time: 20:01)

HDFS - User Commands (acls)

List acl for a file

```
hdfs dfs -getfacl tdata/geneva.csv
```

List the file statistics - (%r - replication factor)

```
hdfs dfs -stat "%r" tdata/geneva.csv
```

Write to hdfs reading from stdin

```
echo "blah blah blah" | hdfs dfs -put - tdataset/tfile.txt  
hdfs dfs -ls -R  
hdfs dfs -cat tdataset/tfile.txt -cat
```

IIT KHARAGPUR NPTEL ONLINE CERTIFICATE COURSE

Similarly, you can get you can get so access controls. So, just like a Unix file system, the Hadoop file system also supports access control, which is you know the owner of a file and then what permissions to the owner has, what permissions to the group has and what permissions to other has. So, it has exactly the same access control model as the UNIX.

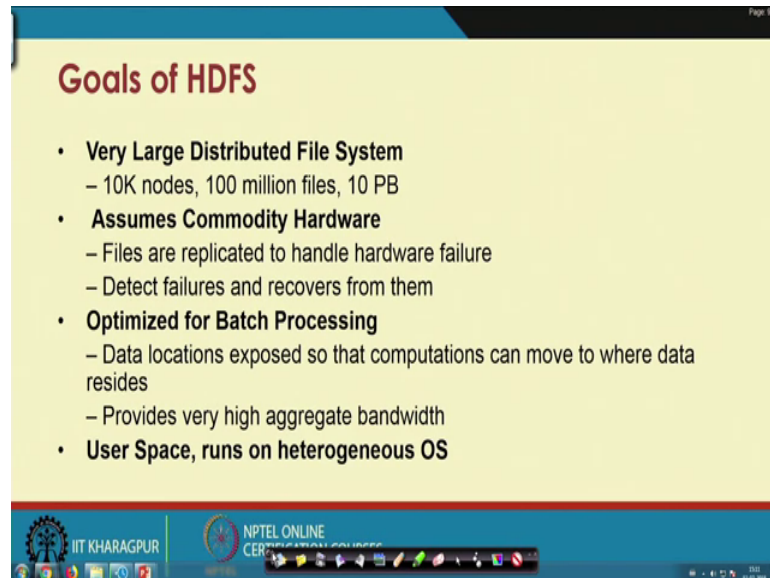
Similarly, you can get some statistics like for a particular file, you can get what is the replication factor, what is the what is the you know creation time, modification time, last access time and so on, using this minus stat command, so again these are very useful commands.

And finally, there are two commands one is this minus put and the other one is minus cat. These commands are useful for you know streaming data. So, minus put allows you to stream data from the std in to directly to a distributed file system without writing it to a local file system.

So, this is very important, because many times you have a very big data, which is generated from some other application. Maybe you are having you have the data in a zip format or some such format and you want to directly put it to HDFS without putting it to the local file system, because your local file system does not have enough space, then you can use these kind of commands.

And cat is the opposite of that, cat will allow you to print or will allow you to show file in a Hadoop distributed file system on to the std out or on to your screen.

(Refer Slide Time: 22:22)



Goals of HDFS

- **Very Large Distributed File System**
 - 10K nodes, 100 million files, 10 PB
- **Assumes Commodity Hardware**
 - Files are replicated to handle hardware failure
 - Detect failures and recovers from them
- **Optimized for Batch Processing**
 - Data locations exposed so that computations can move to where data resides
 - Provides very high aggregate bandwidth
- **User Space, runs on heterogeneous OS**

The slide is part of an NPTEL online course from IIT Kharagpur. The footer includes the IIT Kharagpur logo and the text 'NPTEL ONLINE COURSES'. The slide number 'Page 8/11' is visible in the top right corner.

So, just to recall, as we have discussed the goals of HDFS are so, in order to store very large distributed file something like let us say 10,000 nodes, 100 million files and 10 petabytes of data. So, these kind of statistics you may So, it is not useful to use HDFS for storing a very small amount of data ok.

Moreover, it assumes that you are running this as Hadoop system on commodity hardware. So, hardware failure is the norm and hence files have to be replicated. And the file system is designed in such a manner that it detects failures and recovers from them ok.

Also it is optimized for batch processing that is So, the data locations are exposed, so that computation can move to where the data resides. And when that happens, it provides a very high aggregate bandwidth. So, basically what this means is that when you start a computation your results may start coming slower. So, it is not a low latency data processing, but it is a high throughput that is a large amount of data will be processed in a short amount of time.

And it another design which is also common to Hadoop is that it runs in user space and on heterogeneous operating system. So, you are not a really bound to the system, which on which the whole HDFS system is running.

(Refer Slide Time: 24:30)

Distributed File System

- **Single Namespace for entire cluster**
- **Data Coherency**
 - Write-once-read-many access model
 - Client can only append to existing files
- **Files are broken up into blocks**
 - Typically 128 MB block size
 - Each block replicated on multiple DataNodes
- **Intelligent Client**
 - Client can find location of blocks
 - Client accesses data directly from DataNode

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSE

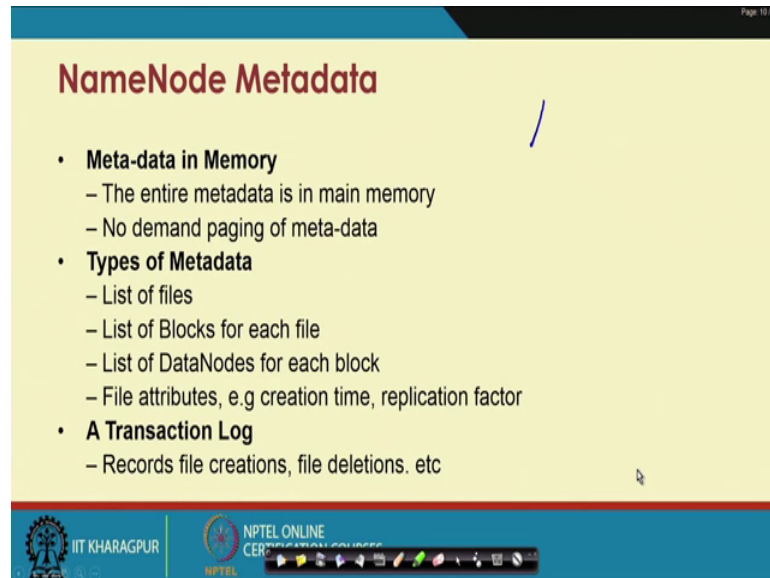
And as we have discussed, so you have only single namespace for the entire cluster. So, everything is under the root directory, which is slash. And you have different subdirectories models just like UNIX file system. And we have also seen that you have write once and read many times access model.

Now, the way this works is that so, mostly you can read you can you can do a contiguous read very fast, you can also do a seek seeking read, which is which is which is slightly slower. And you can you cannot modify an existing file, you can only append to existing files. We will see why this is the case with the architecture. So, files are typically broken up into blocks of big size. So, typically 128 MB is the block size, which is much higher than the standard block size for file systems. And each block is replicated on multiple data nodes.

Now, your HDFS client is directly interacts with the data nodes to access the data. So, this helps in avoiding the bottle neck at the name node level, where only the meta data is stored. So,. So, the metadata is typically much smaller than your entire data, so that there is no single bottleneck. The client has to be intelligent enough to read the locations of the

files from the name node and then directly interact with the data node in order to access those data ok.

(Refer Slide Time: 26:51)

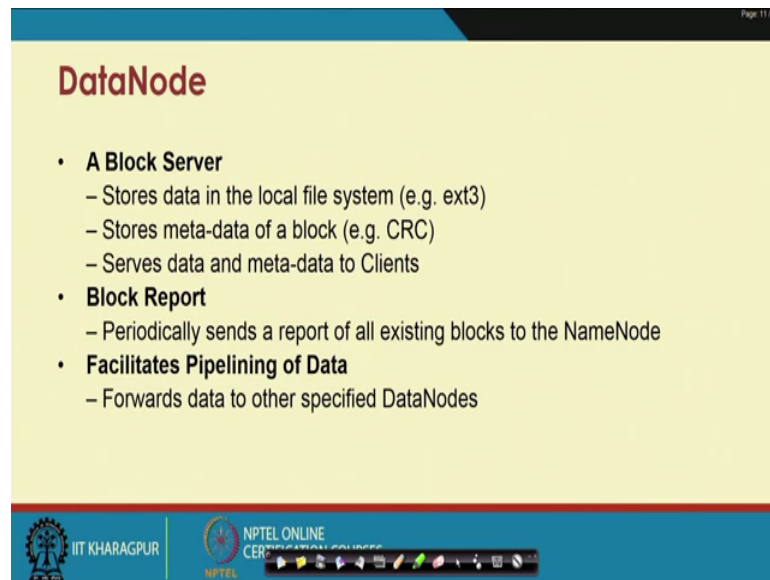


The slide is titled "NameNode Metadata" in a dark red font. It contains three main bullet points, each with sub-points. The first bullet point is "Meta-data in Memory", with sub-points: "The entire metadata is in main memory" and "No demand paging of meta-data". The second bullet point is "Types of Metadata", with sub-points: "List of files", "List of Blocks for each file", "List of DataNodes for each block", and "File attributes, e.g creation time, replication factor". The third bullet point is "A Transaction Log", with a sub-point: "Records file creations, file deletions. etc". The slide has a yellow background and a blue footer. The footer contains the IIT Kharagpur logo, the NPTEL ONLINE logo, and a navigation bar with various icons.

- **Meta-data in Memory**
 - The entire metadata is in main memory
 - No demand paging of meta-data
- **Types of Metadata**
 - List of files
 - List of Blocks for each file
 - List of DataNodes for each block
 - File attributes, e.g creation time, replication factor
- **A Transaction Log**
 - Records file creations, file deletions. etc

So, as discussed already metadata for the; so, the name node stores all the metadata. So, typically it will store a list of files ok. And then it will store for each file a list of blocks. And then it will also store for each block, the list of data nodes on which these blocks is blocks are stored. So, this is the hierarchy in which it will store attributes. Moreover, for each file, it will store attribute such as creation time, replication factor etcetera. Now, this entire metadata will be stored in the memory for the name node. So, typically the server which will host name node is expected to have a high memory ok. And this is accessed by all the applications in the entire cluster ok.

(Refer Slide Time: 28:02)



DataNode

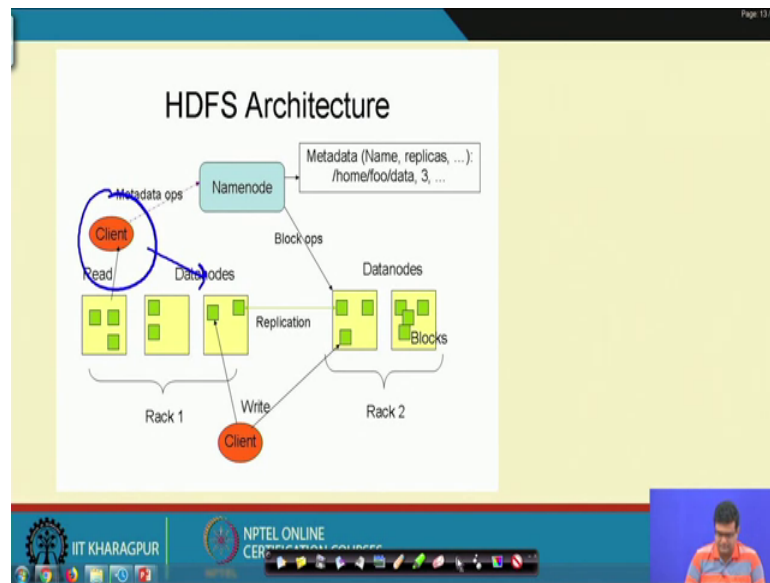
- **A Block Server**
 - Stores data in the local file system (e.g. ext3)
 - Stores meta-data of a block (e.g. CRC)
 - Serves data and meta-data to Clients
- **Block Report**
 - Periodically sends a report of all existing blocks to the NameNode
- **Facilitates Pipelining of Data**
 - Forwards data to other specified DataNodes

IIT KHARAGPUR | NPTEL ONLINE COURSE | NPTEL

And the data node acts as a block server. So, typically a data node would not only store the blocks in its in its local file system ok, but it will also act as a node for processing data in the map reduce framework, so that you can have data local computation.

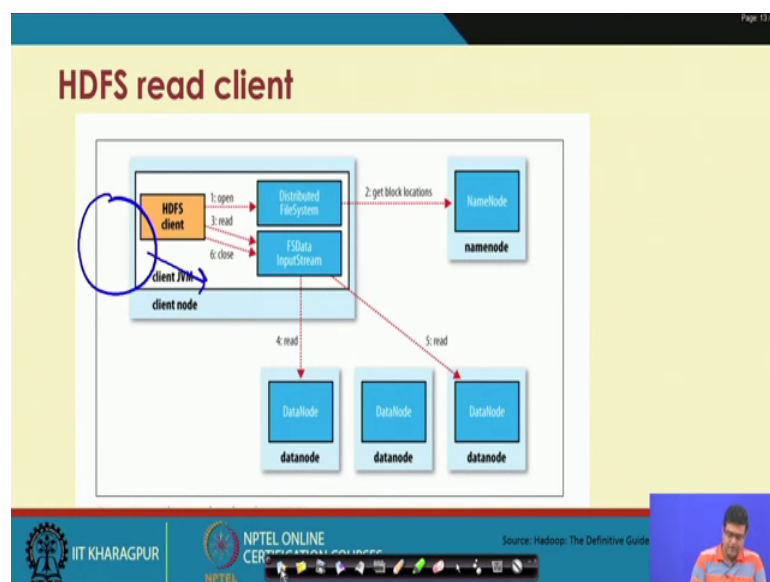
So, so it will store the block data in the local file system. And then it will store some metadata for the block like the check, sum and etcetera. And then as a general rule, it serves the data to the and the meta-data to the clients, which connect to them. And it will also periodically interact with the name node to send the block report to the name node. And the third functionality of the data node is that it facilitates pipelining, which we will see in a minute.

(Refer Slide Time: 29:06)



So, this architecture we have already seen. So, just to make the point here that so, whenever a client whenever a client accesses a so whenever the client accesses a name node and gets the name node gets the metadata for a particular file like home, food, data. And then let us say it is there on the third server. Then it will access that particular data node directly and gets the data.

(Refer Slide Time: 29:53)

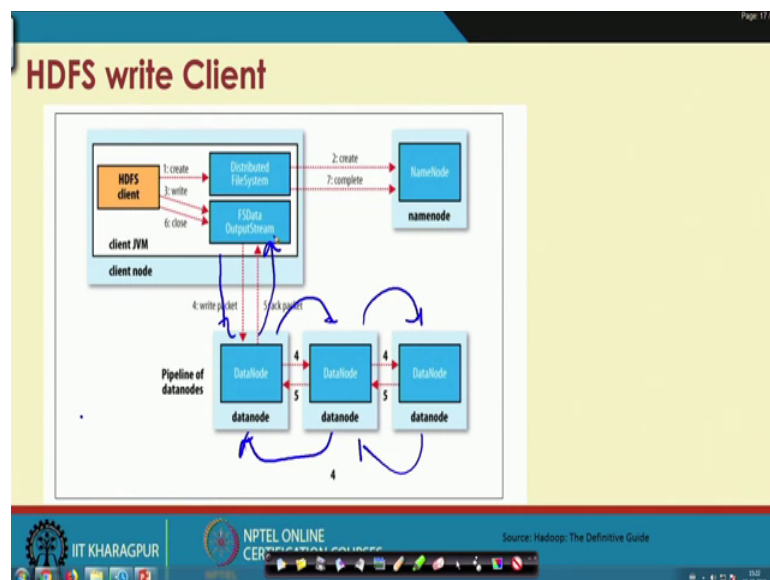


Now, we discuss what how does HDFS read client function. So, a HDFS read client first gets a so, it has to read a particular file. So, it first calls the open function ok, which

accesses the distributed file system and then it access the name node. And it basically results in getting the block locations for this particular file.

So and this block location is now transferred to this HDFS client, which creates this FS input data stream object, where all this block locations are stored. And this object now initiates the reads from this block. So, this object will initiate the reads one by one from the different data nodes and the data nodes will directly supply the data to the read to the client in this machine. And then once it is done with reading all the blocks, it will just issue a close message, which will result in the completion of the read activity.

(Refer Slide Time: 31:23)



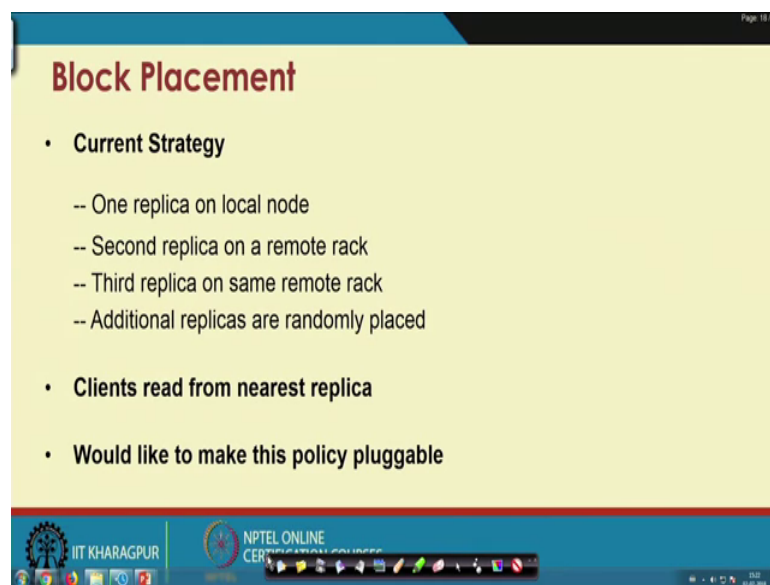
Now, this is simple. On the other hand, the write operation is slightly more complicated; the reason the write operation is more complicated is because when you So, first when you write a new file, so you can either append to an existing file or you can write to a to a new file. So, you call the create function to the name node, the create function then creates some creates some blocks on some data nodes and returns the list of blocks to the FS data output stream object, which is created on the client side.

Then the client writes directly to the data node. So, this gives the data nodes, this gives the address of the data node to which actually three data node, so whatever is the replication factor that many data nodes to the client. So, the client issues a write request to the first data node that it receives. And then that particular data node actually takes care. So, the data nodes they among themselves take care of the replication ok.

So, for example, the first data nodes, now issues write to the second data node. So, the client writes to the first data node, the first data node writes to the second data nodes and the second data node writes to the third data node, if the replication factor is three. And then the acknowledgement comes back from the third to the second, then second to the first, which is finally transferred back to the client. At that point in time, this particular write request is complete.

So, you see the a particular write request takes a lot of time to execute. So, what the file system does or what the write line does rather is it is it issues, it breaks down the data into a different blocks and then it issues the write request to the blocks in parallel and then it the data comes back. And once all the write requests are complete, the client can issue a close request on the file and then the whole write operation is complete.

(Refer Slide Time: 34:12)



Block Placement

- **Current Strategy**
 - One replica on local node
 - Second replica on a remote rack
 - Third replica on same remote rack
 - Additional replicas are randomly placed
- **Clients read from nearest replica**
- **Would like to make this policy pluggable**

IIT KHARAGPUR NPTEL ONLINE COURSES

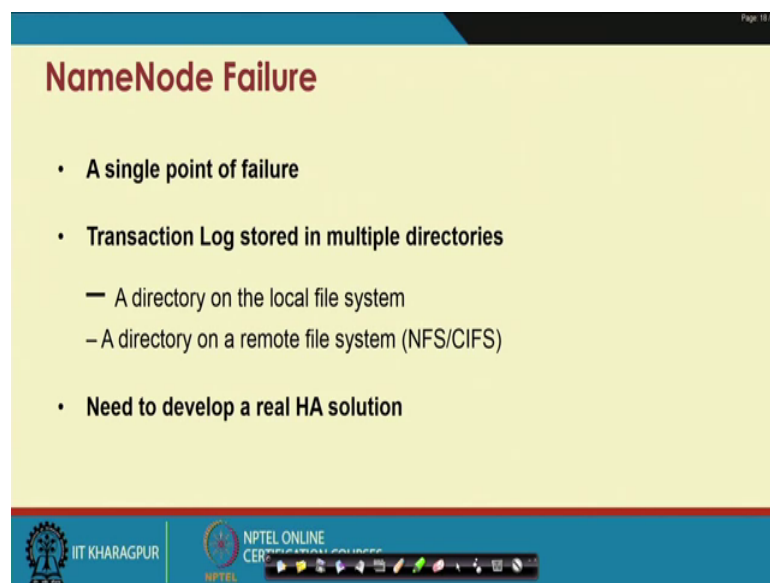
Now, here one important question is how to place the blocks. So, multiple strategies can be used. So, practitioners by trial and error arrived at the following strategies. So, the first, so wherever the data node client is the first write or the first replica happens on a local node, if it is possible ok. So, the first replica of a block is written on to a local node, if possible.

The second replica is happens on a remote rack ok. So, on a; so, typically as we have discussed, the servers are arranged in racks. So, the second replica happens on a rack

different from the node of the first rack. And the third replica happens on the same rack as the second replica.

So, so this is because then there is a rack local write from second to third, but also it ensures that in case of failure of an entire rack, you have the data on at least two racks. So, this is the write placement this is the strategy when you write a data on to a HDFS rack. When you read data, you are your read is typically from the nearest. So, if it tries to do a rack local read and if not possible, it tries to read from the nearest rack.

(Refer Slide Time: 36:20)



The slide is titled "NameNode Failure" in a dark red font. It contains three main bullet points. The first is "A single point of failure". The second is "Transaction Log stored in multiple directories", which is followed by two sub-bullets: "A directory on the local file system" and "A directory on a remote file system (NFS/CIFS)". The third main bullet point is "Need to develop a real HA solution". The slide has a yellow background and a blue footer. The footer contains the IIT Kharagpur logo, the NPTEL ONLINE logo, and a navigation bar with various icons.

So, the single point of failure for the HDFS file system is the name node ok. So, if the name node fails, then so, if any of the data node fails, then so the whole file system or the whole HDFS file system is immune to that, because there are many data nodes, which are there. And any particular block is actually written on to something like free data nodes or whatever is the replication factor for the HDFS. Typically, the replication factor is three.

So, it is unless all three data nodes on which a particular block is existing fail at the same time, which is highly unlikely one can always access a block. But, if the name node fails, then it is a single point of failure. So, the only way at that point in time is to actually switch to a secondary name node yeah. So, ok.

(Refer Slide Time: 37:37)

Data Pipelining

- Client retrieves a **list of DataNodes** on which to place replicas of a block
- Client writes block to the first DataNode
- The first DataNode forwards the data to the next DataNode in the Pipeline
- When all replicas are written, the Client moves on to write the next block in file

IIT KHARAGPUR | NPTEL ONLINE COURSES | NPTEL

And we have already discussed data pipelining.

(Refer Slide Time: 37:44)

Conclusion:

- We have seen:
 - The structure of HDFS.
 - The shell commands.
 - The architecture of HDFS system.
 - Internal functioning of HDFS.

IIT KHARAGPUR | NPTEL ONLINE COURSES | NPTEL | Sourangshu Bhattacharya | Science and Engg. | 24

So, to conclude, today we have discussed the structure of HDFS. And we have discussed the shell commands and we have also discussed the architecture of how the file what are the components and how each of the components interact with each other at the time of read and write. And we have also discussed some of the internal algorithms like the right pipelining algorithm.

(Refer Slide Time: 38:16)

Page 18/18

References:

- Jure Leskovec, Anand Rajaraman, Jeff Ullman. **Mining of Massive Datasets**. 2nd edition. - Cambridge University Press. <http://www.mmds.org/>
- Tom White. **Hadoop: The definitive Guide**. O'Reilly Press.

IIT KHARAGPUR | NPTEL ONLINE COURSES | Sourangshu Bhattacharya | Science and Engg. | 25

So, the references for this is Hadoop The definitive Guide for by Tom White and also the Mining of Massive Datasets by Jure Leskovec.

Thank you.